Efficient algorithms for calculating the probability distribution of the sum of hypergeometric-distributed random variables

Arne Johannssen, Nataliya Chukhrova, Philippe Castagliola

Efficient algorithms for calculating the probability distribution of the sum of hypergeometric-distributed random variables

Arne Johannssen[1] Nataliya Chukhrova[2] Philippe Castagliola[3]

Abstract

In probability theory and statistics, the probability distribution of the sum of two or more independent and identically distributed (i.i.d.) random variables is the convolution of their individual distributions. While convoluting random variables following a binomial, geometric or Poisson distribution is a straightforward procedure, convoluting hypergeometric-distributed random variables is not. The problem is that there is no closed form solution for the probability mass function (p.m.f.) and cumulative distribution function (c.d.f.) of the sum of i.i.d. hypergeometric random variables. To overcome this problem, we propose an approximation for the distribution of the sum of i.i.d. hypergeometric random variables. In addition, we compare this approximation with two classical numerical methods, i.e., convolution and the recursive algorithm by De Pril, by means of an application in Statistical Process Monitoring (SPM). We provide MATLAB codes to implement these three methods for computing the probability distribution of the sum of i.i.d. hypergeometric random variables in an efficient way. The obtained results show that the proposed approximation has remarkable properties and may be helpful in all fields, where the problem of convoluting hypergeometric-distributed random variables occurs. Therefore, the approximation considered in this paper is well suited to make a change over established practices.

- This article presents theoretical bases of three methods for determining the probability distribution of the sum of i.i.d. hypergeometric random variables: (1) direct convolution, (2) recursive algorithm by De Pril, (3) approximation.
- We provide associated MATLAB codes (including context-specific customizations) for direct implementation of these methods and discuss technical aspects and essential details of the tweaks we have made.
- A representative application example in SPM shows that the proposed approximation is considerably simpler in application than both other methods and it ensures a remarkable high accuracy of the results while reducing computational time considerably.

Method details

In Sections 1 and 2, we describe the calculation procedures and relevant formulas that are subsidiary or directly used in the methodology as well as code segments for implementing the methods in MATLAB. In Section 3, we provide the code segment and its description for an application of the proposed methods in the framework of SPM. In Section 4, we propose key criteria for choosing the appropriate method to compute the probability distribution of the sum of i.i.d. hypergeometric-distributed random variables. Conclusions in Section 5 complete the method article.

1 The hypergeometric distribution

Let us assume that X_i, $i = 1, 2, \ldots, m$, are m i.i.d. hypergeometric random variables of parameters (N, n, p), defined on $\{x_{\min} = \max(0, n - N(1-p)), \ldots, x_{\max} = \min(Np, n)\}$, where m is the number of populations, N

[1]University of Hamburg, Hamburg, Germany, arne.johannssen@uni-hamburg.de (corresponding author).
[2]University of Hamburg, Hamburg, Germany, nataliya.chukhrova@uni-hamburg.de.
[3]Université de Nantes & LS2N UMR CNRS 6004, Nantes, France, philippe.castagliola@univ-nantes.fr.

is the population size, n is the sample size and p is the population proportion of some attribute of interest, with $m, N, n \in \mathbb{N}^*$, $N > n$, $p \in [0, 1]$. The p.m.f. and c.d.f. of X_i, $i = 1, 2, \ldots, m$, are then given by

$$f_{X_i}(x) = f_{\mathrm{HYP}}(x|N, n, p) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}} \tag{1.1}$$

$$F_{X_i}(x) = F_{\mathrm{HYP}}(x|N, n, p) = \sum_{j=x_{\min}}^{x} \frac{\binom{M}{j}\binom{N-M}{n-j}}{\binom{N}{n}} \tag{1.2}$$

where $M = \lfloor Np \rfloor$ is the number of units in the population characterized by the attribute of interest.

In the following, the above parameters are referred to as *input variables* or *variables used* (see MATLAB code segment 1 with lines 1–12 in Figure 1). In particular, the parameters m, N, n, p are to initialize (see lines 1–4, parameter p is labeled by `p_0`), while M, $x_{\min}$, $x_{\max}$, $mx_{\min}$, $mx_{\max}$ are computed by MATLAB (see lines 6, 9–12, parameter M is labeled by `M_0`).

```
1  m=50;                  % number of populations
2  N=1000;                % population size
3  n=25;                  % sample size
4  p_0=0.05;              % estimated in-control parameter
5  tau=1;                 % shift parameter tau

6  M_0=floor(N*p_0);      % in-control number of nonconforming units in a population
7  M_1=floor(N*p_0*tau);  % out-of-control number of nonconforming units in a population
8  p_1=M_1/N;             % specified out-of-control parameter

9  x_min=max(0,n-N+M_0);  % minimum possible realization of X_i
10 x_max=min(M_0, n);     % maximum possible realization of X_i

11 s_min=m*x_min;         % minimum possible realization of X
12 s_max=m*x_max;         % maximum possible realization of X
```

Figure 1: Input variables and initializations for a hypergeometric-distributed random variable

In the next section, we determine the probability distribution of the random variable $X = \sum_{i=1}^{m} X_i$ with minimum and maximum possible realization $mx_{\min}$ (labeled by `s_min`, see line 11) and $mx_{\max}$ (labeled by `s_max`, see line 12), respectively, via

(a) convolution,

(b) algorithm by De Pril [1],

(c) approximation by Johannssen et al. [3].

2 Methods for determining the probability distribution of X

2.1 Direct convolution

As the random variable $X = \sum_{i=1}^{m} X_i$ is in general not hypergeometric-distributed (apart from special cases $m = 1$ or $n = N$ or $p \in \{0, 1\}$), we need a practicable way to determine the probability distribution of X. Due to the fact that $X_1, \ldots, X_m$ are i.i.d. random variables, this can be achieved by the direct method of *convolution*. In particular, the probability distribution of the sum of i.i.d. hypergeometric random variables X_i, $i = 1, \ldots, m$, is the convolution of their individual distributions.

The m-fold convolution of f_{X_i} and F_{X_i} can be obtained via following recursive equations using general convolution equations (see e.g., Dickson [2]),

$$f_X(s) = f_{X_i}^{(m)}(s) = \underbrace{f_{X_i} \circledast \cdots \circledast f_{X_i}(s)}_{\times m} = \sum_{j=mx_{\min}}^{s} f_{X_i}^{(m-1)}(s-j) f_{X_i}(j) \tag{2.1}$$

$$F_X(s) = F_{X_i}^{(m)}(s) = \underbrace{F_{X_i} \circledast \cdots \circledast F_{X_i}(s)}_{\times m} = \sum_{j=mx_{\min}}^{s} F_{X_i}^{(m-1)}(s-j) f_{X_i}(j) \tag{2.2}$$

where

$$f_{X_i}^{(0)}(s) = \begin{cases} 1 & \text{for} \quad s = 0 \\ 0 & \text{for} \quad s > 0 \end{cases} \qquad \text{and} \qquad f_{X_i}^{(1)} = f_{X_i}$$

$$F_{X_i}^{(0)}(s) = 1 \qquad \text{and} \qquad F_{X_i}^{(1)} = F_{X_i}$$

and $s \in \{mx_{\min}, \ldots, mx_{\max}\}$. Note that the order of convolution does not matter. Since there are no closed form solutions for f_X and F_X, the *analytical* calculation of f_X and F_X using convolution is impossible.

As for the *numerical* calculation of f_X and F_X with help of (2.1) and (2.2), respectively, code segment 2 (see lines 13–38 in Figure 2) enables for it. In particular, using `for` loops and `if` conditions, the calculation of f_X (labeled by `f_x_c`, line 13) and F_X (labeled by `S_c`, line 14) is carried out as 1-, 2-, ..., m-fold convolution (see lines 21–38), i.e., the algorithm provides results for the sum of $2, 3, \ldots, m$ i.i.d. hypergeometric random variables (see rows 2–m in output vectors `f_x_c` and `S_c` and lines 28–38 in code segment 2) as well as for a single hypergeometric-distributed random variable X_i with help of (1.1) and (1.2) (see row 1 in output vectors `f_x_c` and `S_c` and lines 21–27 in code segment 2). It is worth noting that there is a subsidiary vector `s_vec`, which contains values of s for $1, 2, \ldots, m$ populations (see lines 15–20 in code segment 2).

```
13 f_x_c=zeros(m,s_max-s_min+1);  % vector of exact probabilities (convolution)
14 S_c=zeros(m,s_max-s_min+1);    % vector of exact cumulative probabilities (convolution)

15 s_vec=zeros(m,s_max-s_min+1);  % vector of values of s for i=1,2,...,m
16 for i=1:m
17 for j=1:(i*(x_max-x_min)+1)
18 s_vec(i,j)=i*x_min+j-1;
19 end
20 end

21 for j=1: length(s_vec(1,:))
22 s=s_vec(1,j);
23 if s_vec(1,j)==0 && j==1|| s>0
24 f_x_c(1,j)=hygepdf(s, N, M_0, n);
25 S_c(1,j)=hygecdf(s, N, M_0, n);
26 end
27 end

28 for k=2:m
29 for j=1:length(s_vec(1,:))
30 s=s_vec(k,j);
31 if s_vec(k,j)==0 && j==1|| s>0
32 for i=0:(j-1)
33 f_x_c(k,j)=f_x_c(k,j)+f_x_c(k-1,j-i)*f_x_c(1,i+1);
34 S_c(k,j)=S_c(k,j)+f_x_c(k,i+1);
35 end
36 end
37 end
38 end
```

Figure 2: Numerical calculation of f_X and F_X by means of convolution

2.2 The algorithm by De Pril [1]

To increase the speed of calculation (while keeping a very high accuracy), the *recursive algorithm* by De Pril [1] can be utilized instead of direct convolution for the evaluation of f_X and F_X. In particular, the numerical calculation of f_X and F_X via convolution requires an increased computational effort already for mid-sized values of n ($50 < n < 100$) and m ($100 < m < 1000$), and a very high computational effort for larger values of n ($n \geq 100$) and m ($m \geq 1000$).

The *standard form* of the algorithm by De Pril [1] with respect to the hypergeometric distribution is given by

$$f_X(s) = f_{X_i}^{(m)}(s) = P(X = s) = \begin{cases} (f_{X_i}(0))^m & \text{if } s = 0 \\ \frac{1}{f_{X_i}(0)} \sum_{j=1}^{s} ((m+1)\frac{j}{s} - 1) f_{X_i}(j) P(X = s - j) & \text{if } s \in \mathbb{N}^* \end{cases} \tag{2.3}$$

$$F_X(s) = F_{X_i}^{(m)}(s) = \sum_{j=0}^{s} f_{X_i}^{(m)}(j) \tag{2.4}$$

where X_i, $i = 1, \ldots, m$, are i.i.d. hypergeometric random variables with $f_{X_i}(0) \neq 0$, i.e., $x_{\min} = 0$ ($p \ll 1$) is satisfied. In the case of $x_{\min} > 0$, the algorithm in *shifted form* is to employ, see De Pril [1].

It is worth noting that the algorithm by De Pril [1] is optimal for the computation of f_X and F_X using small values of m, n, p, i.e. $m \leq 100$, $n \leq 50$, $p \leq 0.1$. However, for mid-sized values of m ($100 < m < 1000$), the results of the algorithm by De Pril [1] converge to zero due to the fact that $(f_{X_i}(0))^m = 0$ using standard settings of statistical software, i.e., double-precision floating point numbers (like in MATLAB, `R`,...) instead of variable-precision floating point numbers. On the other hand, the computation of f_X and F_X with variable-precision floating point numbers is very time expensive for $m \geq 1000$ and thus can not be recommended for the researcher/practitioner. Further, higher values of n and/or p lead to single infinitesimal hidden round-off errors, which can raise to substantial errors due to multiplication operations and thus entail unreliable results. This problem in turn stems from standard settings of statistical software.

As for the *numerical* calculation of f_X and F_X with help of (2.3) and (2.4), respectively, the code segment 3 (see lines 39–53 in Figure 3) enables for it. In particular, the calculation of f_X (labeled by `f_x_p`, line 40) and F_X (labeled by `S_p`, line 41) is carried out using `for` loops (see lines 47–53) with initial values $f_{X_i}^{(m)}(0)$ and $F_{X_i}^{(m)}(0)$ (see lines 45–46) and subsidiary values of f_{X_i} (calculated via (1.1)) with respect to a single hypergeometric distributed random variable X_i (see lines 39, 42–44).

```
39 f_x_i=zeros(1,s_max+1);   % vector of hypergeometric probabilities
40 f_x_p=zeros(1,s_max+1);   % vector of exact probabilities (De Pril)
41 S_p=zeros(1,s_max+1);     % vector of exact cumulative probabilities (De Pril)

42 for j=0:x_max
43 f_x_i(1,j+1)=hygepdf(j, N, M_0, n);
44 end

45 f_x_p(1,1)=(f_x_i(1,1))^m;
46 S_p(1,1)=f_x_p(1,1);

47 for s=1:s_max
48 for j=1:s
49 f_x_p(1,s+1)=f_x_p(1,s+1)+(((m+1)*j/s)-1)*f_x_i(1,j+1)*f_x_p(1,s-j+1);
50 end
51 f_x_p(1,s+1)=max(0,f_x_p(1,s+1)/f_x_i(1,1));
52 S_p(1,s+1)=S_p(1,s)+f_x_p(1,s+1);
53 end
```

Figure 3: Numerical calculation of f_X and F_X by means of the algorithm by De Pril [1]

2.3 The approximation by Johannssen et al. [3]

In order to find a remedy in cases with higher values of m, n or p, Johannssen et al. [3] have proposed a new *approximation* of f_X and F_X. This approach is neither based on convolution nor on the algorithm by De Pril [1], but on the approximation of the sum of i.i.d. hypergeometric random variables X_i, $i = 1, \ldots, m$, by a hypergeometric-distributed random variable Z of parameters (mN, mn, p), i.e.,

$$f_X(s) \approx f_{\mathrm{HYP}}(s|mN, mn, p) \tag{2.5}$$

$$F_X(s) \approx F_{\mathrm{HYP}}(s|mN, mn, p) \tag{2.6}$$

Using this approximation, the calculation of f_X and F_X is considerably simplified, and thus the computational time can be reduced to a few seconds while keeping a remarkable high accuracy. Moreover, a comprehensive range of numerical results shows that the higher the values of m, N, n, the better the approximation (see Johannssen et al. [3]). Nonetheless, this method can also be successively used for smaller values of the parameters m, N, n, p resulting in minor deviations from exact results.

As for the *numerical* calculation of f_X and F_X with help of (2.5), (1.1) and (2.6), (1.2), respectively, code segment 4 (see lines 54–59 in Figure 4) enables for it. In particular, the calculation of f_X (labeled by `f_x_a`, line 54) and F_X (labeled by `S_a`, line 55) is in turn carried out using `for` loop (see lines 56–59).

```
54 f_x_a=zeros(1,s_max-s_min+1); % vector of approximate probabilities
55 S_a=zeros(1,s_max-s_min+1);   % vector of approximate cumulative probabilities

56 for j=s_min:s_max;
57 f_x_a(1,j-s_min+1)=hygepdf(j, m*N, m*M_0, m*n);
58 S_a(1,j-s_min+1)=hygecdf(j, m*N, m*M_0, m*n);
59 end
```

Figure 4: Numerical calculation of f_X and F_X by means of the approximation by Johannssen et al. [3]

3 An application example in SPM

In this section, we provide an application example of the proposed methods in SPM for the hypergeometric *np* chart with estimated parameter p_0 (see Johannssen et al. [3]). In particular, we show how to calculate in- and out-of-control measures, i.e., the average run length (ARL) and its standard deviation (SDRL) associated with this control chart, using p.m.f. f_X, which is determined via (1) convolution, (2) the algorithm by De Pril [1], (3) the new approximation by Johannssen et al. [3].

The hypergeometric *np* chart allows for monitoring the number of nonconformings in finite and infinite horizon processes. The chart with estimated parameter is superior to the one with known parameter as well as to its binomial counterpart (i.e., the binomial *np* chart), given the parameter p_0 is unknown (see Johannssen et al. [3]).

The mathematical model underlying the hypergeometric *np* chart for the number of nonconforming units is based on the hypergeometric distribution. Let us assume that Y_i, $i = 1, 2, \ldots, k$, are k Phase II independent random variables corresponding to the number of nonconforming units obtained after sampling *without replacement* n units in a population of size $N > n$, $N, n \in \mathbb{N}^*$, containing an *unknown* proportion $p_1 \in [0, 1]$ of nonconforming units. By definition, Y_i, $i = 1, 2, \ldots, k$, are hypergeometric random variables of parameters (N, n, p_1), defined on $\{y_{\min} = \max(0, n - N(1 - p_1)), \ldots, y_{\max} = \min(Np_1, n)\}$.

Further, let p_0 be the in-control proportion of nonconforming units. Given that p_0 is unknown, it has to be estimated from m Phase I independent random variables X_i, $i = 1, \ldots, m$, corresponding to the

number of nonconforming units obtained after sampling *without replacement* n units in an *in-control* population of size $N > n$ (i.e., containing a proportion $p_0 \in [0,1]$ of nonconforming units). By definition, X_i, $i = 1, \ldots, m$, are hypergeometric random variables of parameters (N, n, p_0) defined on $\{x_{\min} = \max(0, n - N(1-p_0)), \ldots, x_{\max} = \min(Np_0, n)\}$.

An estimator $\hat{p}_0$ of p_0 is given by the best linear unbiased estimator (BLUE)

$$\hat{p}_0 = \frac{1}{mn} \sum_{i=1}^{m} X_i = \frac{X}{mn}$$

(see Johannssen et al. [3]). Since X does in general not follow a hypergeometric distribution, the probability distribution $f_X(x|m, n, N, p_0)$ of X can be achieved by implementing one of the methods proposed in Section 2. In addition, the Shewhart-type control limits of the hypergeometric np chart with estimated parameter p_0 are given by

$$\widehat{\mathrm{LCL}} = \max\left\{0, \left\lceil n\hat{p}_0 - K\sqrt{n\hat{p}_0(1-\hat{p}_0)\frac{N-n}{N-1}} \right\rceil \right\} \tag{3.1}$$

$$\widehat{\mathrm{UCL}} = \left\lfloor n\hat{p}_0 + K\sqrt{n\hat{p}_0(1-\hat{p}_0)\frac{N-n}{N-1}} \right\rfloor \tag{3.2}$$

where $n\widehat{p}_0$ corresponds to the center line $\widehat{\mathrm{CL}}$ and $K > 0$ is a chart parameter.

The (unconditional) ARL, SDRL of the hypergeometric np chart with estimated parameter p_0 are defined by

$$\mathrm{ARL} = \sum_{x=mx_{\min}}^{mx_{\max}} f_X(x|m, n, N, p_0) \left(\frac{1}{\widehat{\theta}}\right) \tag{3.3}$$

$$\mathrm{SDRL} = \sqrt{\mathbb{E}[\mathrm{RL}^2] - \mathrm{ARL}^2} \tag{3.4}$$

with

$$\begin{aligned} \widehat{\theta} = 1 - F_{\mathrm{HYP}}\Bigg(& \overbrace{\left\lfloor \frac{x}{m} + K\sqrt{\frac{x}{m}\left(1 - \frac{x}{mn}\right)\frac{N-n}{N-1}} \right\rfloor}^{\text{conditional UCL}} \Bigg| N, n, p_1 \Bigg) \\ & + F_{\mathrm{HYP}}\Bigg(\underbrace{\left\lceil \frac{x}{m} - K\sqrt{\frac{x}{m}\left(1 - \frac{x}{mn}\right)\frac{N-n}{N-1}} \right\rceil}_{\text{conditional LCL}} - 1 \Bigg| N, n, p_1 \Bigg) \end{aligned} \tag{3.5}$$

and

$$\mathbb{E}[\mathrm{RL}^2] = \sum_{x=mx_{\min}}^{mx_{\max}} f_X(x|m, n, N, p_0) \left(\frac{2 - \widehat{\theta}}{\widehat{\theta}^2}\right) \tag{3.6}$$

where $f_X(x|m, n, N, p_0)$ is the p.m.f. of X, $\widehat{\theta}$ is the probability that the number of nonconforming units in the i-th sample is outside the conditional control limits LCL and UCL (i.e., control limits defined conditionally to $X = x$), and RL is the run length of the hypergeometric np chart with estimated parameter p_0.

Considering $p_1 = p_0$ and $p_1 > p_0$, we obtain in-control and out-of-control performance measures, i.e., $\mathrm{ARL}_0, \mathrm{SDRL}_0$ and $\mathrm{ARL}_1, \mathrm{SDRL}_1$, respectively. Defining $p_1 = \tau p_0$ with $\tau > 1$, the parameters p_0 and τ have to be fixed (see line 4 for p_0, line 5 for τ, and lines 7–8 for calculation of p_1 in code segment 1).

Remark 3.1 *In order to allow adequate comparisons with the performance measures in the known parameter case (given $p_0 = \widehat{p}_0$), the calculation of* ARL *and* SDRL *in the estimated parameter case should be performed under the additional condition of reasonable control limits. Due to the nature of Shewhart-type control limits, the (conditional) probability of a violation of the control limits can be equal to zero either for both cases or only for the estimated parameter case (for some values of x, see* (3.5)*) that results in infinitely large values of* ARL *and* SDRL*. While in the first scenario both charts are equivalent, in the second scenario the performance measures of the hypergeometric np chart with estimated parameter p_0 would be biased. To overcome this bias, we propose to use* $\widehat{\text{UCL}}$ *given by* (3.2) *(which corresponds to* UCL *in the known parameter case) instead of conditional* UCL *(see* (3.5)*) for affected values of x in the calculation of $\widehat{\theta}$. This procedure reduces the bias in an appropriate way by considering $\widehat{\theta} > 0$ instead of $\widehat{\theta} = 0$, i.e., by implementing a reasonable conditional upper control limit* $\text{UCL} < \min(N\widehat{p}_0, n)$ *and* $\text{UCL} < \min(Np_1, n)$*, respectively (see Johannssen et al. [3]).*

As for the *numerical* calculation of the in-control and out-of-control measures, code segment 5 (see lines 60–123 in Figure 5) enables for it. It is worth noting that, for instance, ARL_0 is labeled by `ARL_0_c`, `ARL_0_p` and `ARL_0_a`, where `c`, `p` and `a` stands for convolution, algorithm by De Pril [1] and approximation, respectively. Tagging via `c`/`p`/`a` also holds for $\text{ARL}_1, \text{SDRL}_0, \text{SDRL}_1, \mathbb{E}[\text{RL}_0^2], \mathbb{E}[\text{RL}_1^2]$ in the following. The particular calculation of $\text{ARL}_0, \text{ARL}_1$ (see (3.3), lines 102–104, 108–110) is carried out using `for` loop and four `if` conditions (see lines 72–114) with initial values (see lines 60–62, 66–68) and subsidiary calculation of

- center line $\widehat{\text{CL}}$ and control limits $\widehat{\text{LCL}}$, $\widehat{\text{UCL}}$ (see (3.1), (3.2), lines 80–83, labeled by `CL_Y`, `LCL_Y`, `UCL_Y`, with K as a constant to be fixed in line 73),
- conditional center line CL (see line 74) and control limits LCL, UCL (see (3.5), lines 75–77),
- adjusted conditional UCL following the approach of reasonable control limits (see lines 78–79 for initial values of conditional UCL regarding in-control and out-of-control scenarios, labeled by `UCL_0` and `UCL_1`, as well as both `if` conditions in lines 84–86 and 87–89),
- estimators $\widehat{\theta}_0$ and $\widehat{\theta}_1$ via (3.5) (see lines 90–91, labeled by `teta_0` and `teta_1`, as well as lines 92–96 and 97–101 for exclusion of $\widehat{\theta}_0 = 0$ and $\widehat{\theta}_1 = 0$, caused by standard settings of statistical software, from calculations with help of dummy variables c_0 and c_1),

while the calculation of $\text{SDRL}_0, \text{SDRL}_1$ (see (3.4), lines 115–117, 118–120) is provided afterwards under subsidiary calculation of expected values $\mathbb{E}[\text{RL}_0^2]$ and $\mathbb{E}[\text{RL}_1^2]$ via (3.6) (see lines 105–107 and 111–113, labeled by `E_L2_0_c`, `E_L2_0_p`, `E_L2_0_a` and `E_L2_1_c`, `E_L2_1_p`, `E_L2_1_a` with initial values given in lines 63–65, 69–71). Code segment 5 ends with output vectors `r_c`, `r_p`, `r_a` (see lines 121–123), which provide the values of $\text{ARL}_0, \text{SDRL}_0$, $\text{ARL}_1, \text{SDRL}_1$ regarding convolution, algorithm by De Pril [1] and approximation.

```
ARL_0_c=0;
ARL_0_p=0;
ARL_0_a=0;

E_L2_0_c=0;
E_L2_0_p=0;
E_L2_0_a=0;

ARL_1_c=0;
ARL_1_p=0;
ARL_1_a=0;

E_L2_1_c=0;
E_L2_1_p=0;
E_L2_1_a=0;

for j=s_min:s_max

K=3;                        % chart parameter
CL=j/m;                     % conditional center line
Var_X=j*(1-j/(m*n))*(N-n)/(m*(N-1));
LCL=CL-K*sqrt(Var_X);       % conditional lower control limit
UCL=CL+K*sqrt(Var_X);       % conditional upper control limit
UCL_0=UCL;                  % conditional UCL for in-control performance
UCL_1=UCL;                  % conditional UCL for out-of-control performance

CL_Y=n*p_0;                 % center line
Var_Y=n*p_0*(1-p_0)*(N-n)/(N-1);
LCL_Y=CL_Y-K*sqrt(Var_Y);  % lower control limit
UCL_Y=CL_Y+K*sqrt(Var_Y);  % upper control limit

if UCL>=min(M_0, n) && UCL_Y<min(M_0, n)
UCL_0=UCL_Y;                % adjusted conditional UCL for in-control performance
end

if UCL>=min(M_1, n) && UCL_Y<min(M_1, n)
UCL_1=UCL_Y;                % adjusted conditional UCL for out-of-control performance
end

teta_0=1-hygecdf(floor(UCL_0), N, M_0, n)+ hygecdf((ceil(LCL)-1), N, M_0, n);
teta_1=1-hygecdf(floor(UCL_1), N, M_1, n)+ hygecdf((ceil(LCL)-1), N, M_1, n);

if teta_0==0 && UCL_0<min(M_0, n)
c_0=0;
else
c_0=1;
end

if teta_1==0 && UCL_1<min(M_1, n)
c_1=0;
else
c_1=1;
end

ARL_0_c=ARL_0_c+f_x_c(m,j-s_min+1)*(teta_0^(-1*c_0))*c_0;
ARL_0_p=ARL_0_p+f_x_p(1,j+1)*(teta_0^(-1*c_0))*c_0;
ARL_0_a=ARL_0_a+f_x_a(1,j-s_min+1)*(teta_0^(-1*c_0))*c_0;
E_L2_0_c=E_L2_0_c+f_x_c(m,j-s_min+1)*(2-teta_0)/(teta_0^(2*c_0))*c_0;
E_L2_0_p=E_L2_0_p+f_x_p(1,j+1)*(2-teta_0)/(teta_0^(2*c_0))*c_0;
E_L2_0_a=E_L2_0_a+f_x_a(1,j-s_min+1)*(2-teta_0)/(teta_0^(2*c_0))*c_0;

ARL_1_c=ARL_1_c+f_x_c(m,j-s_min+1)*(teta_1^(-1*c_1))*c_1;
ARL_1_p=ARL_1_p+f_x_p(1,j+1)*(teta_1^(-1*c_1))*c_1;
ARL_1_a=ARL_1_a+f_x_a(1,j-s_min+1)*(teta_1^(-1*c_1))*c_1;
E_L2_1_c=E_L2_1_c+f_x_c(m,j-s_min+1)*(2-teta_1)/(teta_1^(2*c_1))*c_1;
E_L2_1_p=E_L2_1_p+f_x_p(1,j+1)*(2-teta_1)/(teta_1^(2*c_1))*c_1;
E_L2_1_a=E_L2_1_a+f_x_a(1,j-s_min+1)*(2-teta_1)/(teta_1^(2*c_1))*c_1;

end

SDRL_0_c=sqrt(E_L2_0_c-ARL_0_c^2);
SDRL_0_p=sqrt(E_L2_0_p-ARL_0_p^2);
SDRL_0_a=sqrt(E_L2_0_a-ARL_0_a^2);

SDRL_1_c=sqrt(E_L2_1_c-ARL_1_c^2);
SDRL_1_p=sqrt(E_L2_1_p-ARL_1_p^2);
SDRL_1_a=sqrt(E_L2_1_a-ARL_1_a^2);

r_c=[min(ARL_0_c, inf), min(SDRL_0_c, inf), min(ARL_1_c, inf), min(SDRL_1_c, inf)]
r_p=[min(ARL_0_p, inf), min(SDRL_0_p, inf), min(ARL_1_p, inf), min(SDRL_1_p, inf)]
r_a=[min(ARL_0_a, inf), min(SDRL_0_a, inf), min(ARL_1_a, inf), min(SDRL_1_a, inf)]
```

Figure 5: An application example in SPM: Calculation of in- and out-of-control measures ARL and SDRL for the hypergeometric np chart with estimated parameter p_0, using f_X determined via (1) convolution, (2) the algorithm by De Pril [1], (3) the approximation by Johannssen et al. [3]

As for numerical results regarding, for instance, the in-control measures of the hypergeometric np chart with estimated parameter p_0, we provide in the Appendix Tables 3–9, which contain the values of $\mathrm{ARL}_0, \mathrm{SDRL}_0$ for $K = 3$ and various combinations of parameters N, n, m, p_0:

$$
\begin{aligned}
N &\in \{100, 200, 500, 1000, 2000, 5000, 10000\} && \text{(see Tables 3–9)} \\
n &\in \{25, 50, 75, 100\} && \text{(see blocks in Tables 3–9)} \\
m &\in \{10, 20, 50, 100, 200, 1000\} && \text{(see columns in Tables 3–9)} \\
p_0 &\in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\} && \text{(see rows in Tables 3–9)}
\end{aligned}
$$

In particular, the first row of each cell of Tables 3–9 shows the ordered pair $(\mathrm{ARL}_0, \mathrm{SDRL}_0)$ calculated via convolution, while the ordered pair $(\mathrm{ARL}_0, \mathrm{SDRL}_0)$ in each second row is calculated via the approximation by Johannssen et al. [3]. The values of $\mathrm{ARL}_0, \mathrm{SDRL}_0$ calculated via convolution correspond to the respective values calculated via the algorithm by De Pril [1], given $N \geq 200$, $n \leq 50$, $m \leq 100$, $p_0 \leq 0.1$. Deviant values of parameters N, n, m, p_0 lead to unreliable results, which is why the respective results obtained by this method (i.e., the algorithm by De Pril [1]) are not given in Tables 3–9.

It is worth noting that in the case of a total inspection of the population (i.e., $N = n = 100$), $\mathrm{ARL}_0, \mathrm{SDRL}_0$ turn out to be infinitely large due to the fact that the type I error is equal to zero. In addition, for $N \in \{100, 200\}$ and $p_0 \in \{0.01, 0.02\}$, the values of ARL_0 and SDRL_0 can in turn be infinitely large (see Tables 3–4). The reason for these exceptions is given by the fact that these parameter combinations do not ensure reasonable control limits and thus the (conditional) probability of a violation of the control limits is equal to zero for some values of x in the estimated parameter case as well as throughout in the known parameter case (see also Johannssen et al. [3]).

The results of Tables 3–9 can be summarized as follows: The values of the in-control measures calculated via the approximation by Johannssen et al. [3] (ARL_0^a, SDRL_0^a) are throughout approximately equal to the respective exact in-control measures obtained via convolution (ARL_0^c, SDRL_0^c). In particular, we observe

- no tendency for a systematic deviation ($\mathrm{ARL}_0^c - \mathrm{ARL}_0^a$, $\mathrm{SDRL}_0^c - \mathrm{SDRL}_0^a$) to one direction regarding n,
- a negative tendency regarding p_0 for smaller values of N ($N \leq 100$), and
- a positive tendency regarding m and smaller values of N.

For increasing values of N, n, m or p_0, the difference between exact and approximate values for in-control measures decreases in a monotonic way and seems to converge to zero. For instance, the maximum of the relative deviation regarding ARL_0 $\left(\frac{|\mathrm{ARL}_0^c - \mathrm{ARL}_0^a|}{\mathrm{ARL}_0^c} \cdot 100\%\right)$ is 2.21% for $N = 100$ and 0.07% for $N = 10000$. We mostly observe relative deviations regarding ARL_0 that are $< 0.1\%$ for $N \leq 1000$ and $< 0.05\%$ for $N > 1000$. These negligible deviations between exact and approximate results are also confirmed when considering out-of-control measures, see Johannssen et al. [3]. That is, the new approximation is preferable for calculations due to considerable reductions of computational time, in particular for larger parameter values.

4 How to choose the appropriate method

In this section, we propose three key criteria for choosing the appropriate method to compute the probability distribution of the sum of i.i.d. hypergeometric-distributed random variables:

1. accuracy of results
2. computational effort
3. grade of simplicity

As there is mostly a trade-off between the satisfaction of these criteria, the "right" choice is dependent on various problem-based factors and up to the practitioner/researcher, and therefore, often a subjective decision. For instance, when cost aspects due to possible inaccuracies of the obtained results are more crucial than computational effort aspects or vice versa, this may lead to different final decisions. On the other hand, there are often real-life situations, where a decision must be made at short notice, i.e., without weighting of single criteria. In the following, we provide appropriate solutions for both these possible settings, based on comprehensive numerical analyzes, in terms of detailed descriptions of considered criteria with regard to single methods (see Table 1) as well as of facilitated guidelines for quick implementations (see decision tree in Figure 6).

Considering Table 1, the practitioner can use color coded ranking, i.e., green as the first choice, yellow as the second choice and red as the third choice, for decision making in single categories under consideration of the magnitude of the parameters m, n, p, N, i.e., low/middle/high. In addition, there are detailed comments (see also the legend in Table 2 related to the entries of Table 1) on the effects of the magnitude of the parameters, which can vary between extremely low and extremely high for the driving parameters (highlighted in bold type) or they are stated in the way "minor/increasing/decreasing" regarding parameters that are less important in a particular category. Finally, there are entries like "exact", "unreliable" or "(minor) deviations" with respect to exact results, inappropriate methods and approximate results in the category "accuracy of results" (see also detailed explanations in Table 2).

Example 4.1
Given $m = 100$, $n = 50$, $p = 0.05$, $N = 2000$, the practitioner could choose

- *the algorithm by De Pril [1], as it provides exact results and a reduced computational effort/grade of complexity in comparison to convolution, or*
- *the approximation, as it ensures an extremely low computational effort while providing a very high grade of simplicity combined with an extremely high accuracy of the approximate results (near to exact results due to relative deviations $< 0.05\%$).*

parameter		accuracy of results			computational effort			grade of simplicity		
		convolution	De Pril	approximation	convolution	De Pril	approximation	convolution	De Pril	approximation
m	low ($m \leq 100$)	exact	exact	minor deviations	**low/middle**	**very low**	**extremely low**	**low**	**middle**	**very high**
	middle ($100 < m < 1000$)	exact	unreliable	increasing	**high**	**low**	**very low**	**very low**	**low**	**very high**
	high ($m \geq 1000$)	exact	unreliable	increasing	**very high**	**middle**	**low**	**extremely low**	**very low**	**very high**
n	low ($n \leq 50$)	exact	exact	minor deviations	**low/middle**	**minor**	**minor**	minor	minor	very high
	middle ($50 < n < 100$)	exact	unreliable	increasing	**high**	**increasing**	**minor**	decreasing	slightly decreasing	very high
	high ($n \geq 100$)	exact	unreliable	increasing	**very high**	**increasing**	**minor**	decreasing	slightly decreasing	very high
p	low ($p \leq 0.1$)	exact	exact	minor deviations	minor	minor	minor	minor	minor	very high
	high ($p > 0.1$)	exact	unreliable	increasing	increasing	slightly increasing	minor	decreasing	slightly decreasing	very high
N	low ($N < 200$)	**exact**	**unreliable**	**high** (max. dev. $\approx 2\%$)	minor	minor	minor	minor	minor	very high
	middle ($200 \leq N < 1000$)	**exact**	**exact**	**very high** (dev. $< 0.1\%$)	increasing	slightly increasing	minor	decreasing	slightly decreasing	very high
	high ($N \geq 1000$)	**exact**	**exact**	**extr. high** (dev. $< 0.05\%$)	increasing	slightly increasing	minor	decreasing	slightly decreasing	very high

Table 1: Color coded ranking for the choice of the appropriate method

Denomination	Explanation
dev.	relative deviations of $x\%$ between approximate and exact results
exact	exact results
increasing (accuracy)	decreasing deviations between approximate and exact results as a result of increasing magnitude of the respective parameter
minor deviations	approximate results with minor deviations from the exact results
unreliable	unreliable results due to standard settings of statistical software
(slightly) increasing (effort)	increasing computational effort as a result of increasing magnitude of respective parameter
minor (effort)	minor input value of respective parameter regarding computational effort
(slightly) decreasing (simplicity)	increasing complexity of computational calculation as a result of an increasing magnitude of respective parameter
minor (simplicity)	minor input value of respective parameter regarding complexity of computational calculation

Table 2: Legend for Table 1

Considering the decision tree in Figure 6, the practitioner can follow the facilitated scheme to obtain a quick solution for specified parameters m, n, p, N. In particular, the decision tree is based on a standardized questionnaire with respect to the magnitude of parameters m, n, p, N (low vs. not low) and implements the most important limiting cases regarding single methods in compliance with conditions on high accuracy of the results and on an adequate computational effort. Following the scheme provided by Figure 6, the practitioner would choose

1. the approximation when there are higher values of m and/or n,
2. convolution when the values of m, n are low in combination with a higher value of p or in combination with low values of p, N, and
3. the algorithm by De Pril [1] when there are low values of m, n, p and higher values of N.

As for Example 4.1, consulting Figure 6 would lead to the algorithm by De Pril [1]. But, when one of the parameters m or n increases, one would choose the approximation. Convolution should be chosen, for instance, when p exceeds 0.1.

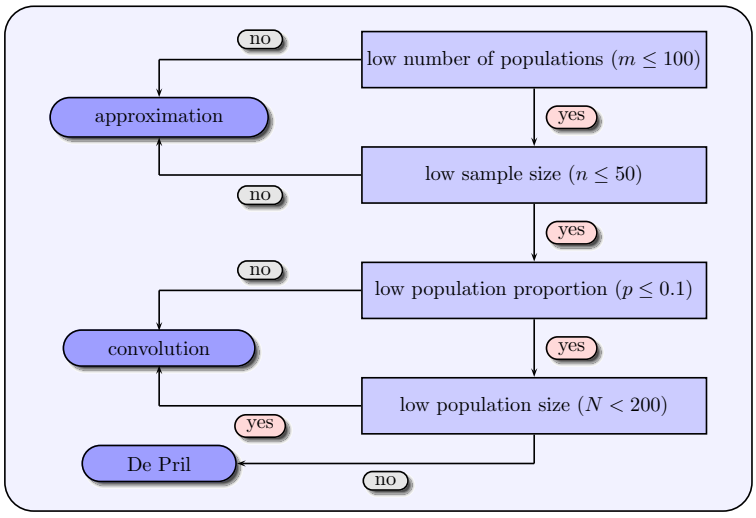

Figure 6: Decision tree for the choice of the appropriate method

5 Conclusions

In this paper, we have shown how to implement three methods – direct convolution, the algorithm by De Pril [1], and the new approximation by Johannssen et al. [3] – for computing the probability mass function of $X = \sum_{i=1}^{m} X_i$ with i.i.d. random variables $X_i \sim \mathrm{HYP}(N, n, p)$, $i = 1, \ldots, m$. While the computational effort using convolution is very extensive for combinations of larger values of m and n, multiplication operations in the framework of the algorithm by De Pril [1] entail unreliable results already for mid-sized values of m, n, p due to hidden round-off errors, and thus cannot be recommended for the researcher/practitioner. In order to get a feasible solution that leads to a considerably faster calculation and simultaneously ensures a high accuracy of the results, we have also considered the new approximation for the probability distribution of X introduced by Johannssen et al. [3]. By using this approximation the time required for the computation of f_X and F_X is reduced to a few seconds while keeping a remarkable high accuracy with only negligible deviations compared to the exact distribution obtained via convolution. Such remarkable properties of the new approximation may be helpful for numerous applications, for instance in SPM or quantitative risk management (e.g., for aggregation of credit, insurance, market, and operational risks, see Klugman et al. [4]).

We have performed an application example in SPM, where we have calculated in-control measures for the hypergeometric np chart with estimated parameter p_0 by means of p.m.f. f_X determined via the above three methods. The results illustrate that in-control measures computed via exact convolution or the algorithm by De Pril [1] are throughout approximately equal to the respective measures calculated via the new approximation with only negligible relative deviations (on average $< 0.1\%$). As the absolute difference between exact and approximate measures converges to zero and the computational effort required for convolution increases enormously for larger values of N, n, m, p_0, the new approximation is clearly preferable for mid-sized and large parameter values. Only in rare cases with lower parameter values, where slightly increased relative deviations (up to about 2%) between exact and approximate results are considered as inadmissible, exact methods are to prefer. On the other hand, if such small deviations are considered as permissible, the new approximation can also be successfully used for smaller values of the parameters m, N, n, p_0 and the practitioner/researcher can take advantage of the considerably simplified handling of the approximation.

Based on comprehensive numerical analyzes, we have summarized all the advantages and limitations of the discussed three methods in schematic form under consideration of the magnitude of the parameters and three key criteria: (1) accuracy of results, (2) computational effort and (3) grade of simplicity. Furthermore, a color coded ranking reflects linked characteristics of the proposed methods in an illustrative way and provides the complete picture needed for an appropriate decision making. By a subjective weighting of the key criteria, the practitioner is more flexible in choosing a proper method. On the other hand, when a quick decision is necessary, a facilitated scheme by means of a decision tree can be used, where the practitioner obtains the adequate method by responding up to four questions regarding the magnitude of the parameters m, n, p, N. The decision tree reflects both limitations of the approximation for smaller parameter values and its advantages for larger parameter values.

Acknowledgments:
The authors would like to thank three anonymous reviewers for their valuable feedback and suggestions, which were important and helpful to improve the paper.

References

[1] De Pril, N. (1985): Recursions for convolutions of arithmetic distributions. *ASTIN Bulletin*, **15**(2), pp. 135–139.

[2] Dickson, D. C. M. (2016): *Insurance Risk and Ruin*. 2nd edition, Cambridge University Press.

[3] Johannssen, A., N. Chukhrova & P. Castagliola (2021): The performance of the hypergeometric *np* chart with estimated parameter. *European Journal of Operational Research*, in press.

[4] Klugman, S. A., H. H. Panjer & G. E. Willmot (2019): *Loss Models: From Data to Decisions*. 5th edition, Wiley Series in Probability and Statistics.

Appendix: Tables of in-control performance

Table 3: Exact and approximate (ARL_0, $SDRL_0$) for the hypergeometric np chart with $N = 100$, $n \in \{25, 50, 75, 100\}$, $p_0 \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$, $m \in \{10, 20, 50, 100, 200, 1000\}$, $K = 3$

p_0	$m = 10$	$m = 20$	$m = 50$	$m = 100$	$m = 200$	$m = 1000$
			$n = 25$			
0.01	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.02	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.05	(729.6, 1199.6)	(798.8, 1238.5)	(898.0, 1288.3)	(987.8, 1326.0)	(1094.6, 1361.8)	(1330.9, 1409.4)
	(729.8, 1199.7)	(799.3, 1238.7)	(898.8, 1288.7)	(989.0, 1326.5)	(1096.1, 1362.2)	(1332.2, 1409.5)
0.10	(3003.9, 63672.0)	(958.3, 4784.4)	(600.6, 1368.5)	(491.7, 656.2)	(484.7, 489.8)	(486.0, 485.5)
	(2937.5, 61362.8)	(949.9, 4647.0)	(597.8, 1355.8)	(491.2, 650.2)	(484.8, 489.5)	(486.0, 485.5)
0.15	(1664.5, 7384.0)	(985.4, 2945.6)	(608.0, 1135.0)	(536.6, 670.8)	(532.5, 539.2)	(535.0, 534.5)
	(1648.7, 7301.5)	(979.5, 2902.8)	(606.1, 1126.2)	(536.2, 666.4)	(532.6, 538.9)	(535.0, 534.5)
0.20	(467.1, 627.2)	(538.5, 684.7)	(603.0, 731.7)	(682.0, 773.3)	(747.1, 797.0)	(812.3, 812.8)
	(467.7, 627.7)	(539.4, 685.4)	(604.1, 732.4)	(683.0, 773.8)	(747.9, 797.2)	(812.4, 812.8)
			$n = 50$			
0.01	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.02	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.05	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.10	(647.2, 791.6)	(741.2, 821.5)	(812.7, 837.3)	(837.0, 841.3)	(842.4, 842.0)	(842.6, 842.1)
	(648.4, 792.0)	(742.2, 821.8)	(813.2, 837.4)	(837.2, 841.3)	(842.4, 842.0)	(842.6, 842.1)
0.15	(393.5, 443.4)	(405.7, 437.0)	(391.1, 418.2)	(363.6, 393.2)	(331.5, 360.0)	(274.0, 282.3)
	(393.9, 443.5)	(405.8, 436.9)	(390.8, 417.9)	(363.2, 392.8)	(331.0, 359.5)	(273.8, 282.0)
0.20	(375.2, 503.5)	(374.7, 462.3)	(416.9, 545.0)	(448.1, 606.4)	(491.3, 680.9)	(684.6, 919.9)
	(375.8, 504.1)	(375.0, 462.8)	(417.2, 545.7)	(448.6, 607.4)	(492.1, 682.1)	(686.1, 921.5)
			$n = 75$			
0.01	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.02	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.05	(909.0, 1293.3)	(1038.8, 1344.3)	(1218.5, 1391.9)	(1334.2, 1409.7)	(1399.4, 1415.5)	(1417.0, 1416.5)
	(910.0, 1293.8)	(1040.2, 1344.7)	(1220.1, 1392.3)	(1335.5, 1409.9)	(1399.9, 1415.5)	(1417.0, 1416.5)
0.10	(615.2, 1433.4)	(495.0, 697.9)	(485.4, 486.7)	(486.0, 485.5)	(486.0, 485.5)	(486.0, 485.5)
	(613.0, 1423.3)	(494.4, 691.3)	(485.4, 486.6)	(486.0, 485.5)	(486.0, 485.5)	(486.0, 485.5)
0.15	(533.2, 995.4)	(512.4, 665.3)	(531.5, 543.8)	(534.7, 534.5)	(535.0, 534.5)	(535.0, 534.5)
	(533.6, 993.5)	(512.6, 663.3)	(531.6, 543.3)	(534.7, 534.5)	(535.0, 534.5)	(535.0, 534.5)
0.20	(423.1, 568.4)	(512.0, 644.4)	(614.1, 719.2)	(708.3, 769.5)	(777.4, 798.8)	(813.4, 813.0)
	(423.9, 569.1)	(513.1, 645.2)	(615.3, 720.0)	(709.5, 770.1)	(778.1, 799.1)	(813.4, 813.0)
			$n = 100$			
0.01	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.02	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.05	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.10	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.15	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.20	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)

Table 4: Exact and approximate (ARL_0, SDRL_0) for the hypergeometric np chart with $N = 200$, $n \in \{25, 50, 75, 100\}$, $p_0 \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$, $m \in \{10, 20, 50, 100, 200, 1000\}$, $K = 3$

p_0	$m = 10$	$m = 20$	$m = 50$	$m = 100$	$m = 200$	$m = 1000$
			$n = 25$			
0.01	(49.8, 62.7)	(59.6, 65.1)	(65.7, 65.8)	(66.3, 65.8)	(66.3, 65.8)	(66.3, 65.8)
	(49.9, 62.8)	(59.7, 65.1)	(65.7, 65.8)	(66.3, 65.8)	(66.3, 65.8)	(66.3, 65.8)
0.02	(1229.3, 3195.1)	(729.4, 2392.5)	(268.1, 1108.9)	(165.7, 368.1)	(155.9, 161.2)	(155.8, 155.3)
	(1227.7, 3192.9)	(726.8, 2387.4)	(266.8, 1102.4)	(165.5, 364.5)	(155.9, 160.9)	(155.8, 155.3)
0.05	(1222.7, 29601.6)	(468.0, 1851.0)	(298.0, 601.0)	(279.9, 322.1)	(290.7, 295.3)	(296.5, 296.0)
	(1211.4, 28091.1)	(466.1, 1831.0)	(297.6, 597.8)	(280.0, 321.4)	(290.7, 295.3)	(296.5, 296.0)
0.10	(996.6, 7421.0)	(498.4, 1352.2)	(366.2, 640.7)	(299.7, 517.1)	(236.9, 371.9)	(189.5, 189.7)
	(990.1, 7284.9)	(497.0, 1341.7)	(365.7, 639.6)	(299.1, 516.0)	(236.5, 370.8)	(189.5, 189.7)
0.15	(1046.3, 5081.1)	(602.5, 1397.6)	(457.9, 728.7)	(388.2, 615.7)	(314.8, 488.6)	(225.7, 234.0)
	(1041.2, 5026.2)	(601.2, 1389.3)	(457.4, 727.6)	(387.7, 615.0)	(314.3, 487.6)	(225.6, 233.7)
0.20	(731.8, 1750.3)	(603.9, 1131.7)	(501.1, 794.7)	(422.8, 619.2)	(360.2, 442.1)	(331.6, 331.2)
	(731.5, 1748.0)	(603.4, 1129.5)	(500.5, 793.5)	(422.3, 617.8)	(359.9, 441.0)	(331.6, 331.1)
			$n = 50$			
0.01	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.02	(233.9, 273.1)	(249.9, 276.4)	(276.3, 280.0)	(280.4, 280.3)	(280.9, 280.4)	(280.9, 280.4)
	(234.1, 273.1)	(250.0, 276.5)	(276.4, 280.0)	(280.4, 280.3)	(280.9, 280.4)	(280.9, 280.4)
0.05	(2986.3, 48816.3)	(994.9, 4910.8)	(592.2, 1353.6)	(425.9, 767.3)	(370.8, 416.9)	(365.3, 364.8)
	(2954.7, 48065.4)	(990.5, 4840.4)	(590.8, 1349.2)	(425.2, 763.7)	(370.7, 415.7)	(365.3, 364.8)
0.10	(700.7, 1292.6)	(699.2, 1253.8)	(566.6, 947.6)	(476.5, 676.0)	(429.1, 473.8)	(418.5, 418.0)
	(700.8, 1292.5)	(698.7, 1252.8)	(565.9, 945.6)	(476.0, 674.1)	(428.9, 472.9)	(418.5, 418.0)
0.15	(412.7, 552.2)	(427.6, 541.1)	(429.4, 530.3)	(468.5, 578.1)	(521.6, 635.1)	(669.1, 749.5)
	(412.8, 552.2)	(427.7, 541.1)	(429.6, 530.5)	(468.8, 578.5)	(522.0, 635.5)	(669.5, 749.7)
0.20	(376.6, 503.9)	(401.4, 487.4)	(438.7, 486.0)	(466.4, 492.4)	(491.3, 500.9)	(507.2, 506.7)
	(376.8, 504.1)	(401.6, 487.5)	(438.9, 486.1)	(466.6, 492.4)	(491.5, 501.0)	(507.2, 506.7)
			$n = 75$			
0.01	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.02	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.05	(4525.9, 13629.3)	(3006.1, 10395.1)	(1328.8, 3856.4)	(1190.5, 1570.5)	(1268.7, 1345.7)	(1357.5, 1357.7)
	(4515.1, 13609.2)	(2994.1, 10364.3)	(1325.7, 3829.8)	(1190.8, 1564.1)	(1269.3, 1345.8)	(1357.6, 1357.7)
0.10	(474.6, 676.7)	(502.9, 662.0)	(482.6, 594.8)	(442.1, 518.5)	(401.0, 438.6)	(371.7, 371.3)
	(474.8, 676.7)	(503.0, 661.8)	(482.3, 594.3)	(441.8, 518.0)	(400.8, 438.1)	(371.7, 371.3)
0.15	(351.9, 410.4)	(378.6, 419.6)	(412.2, 437.0)	(428.0, 445.3)	(440.4, 453.2)	(470.9, 473.3)
	(352.2, 410.5)	(378.8, 419.7)	(412.4, 437.1)	(428.1, 445.4)	(440.4, 453.2)	(471.0, 473.3)
0.20	(325.4, 390.0)	(369.2, 428.9)	(423.4, 480.7)	(471.3, 521.7)	(523.0, 557.2)	(590.7, 592.6)
	(325.6, 390.1)	(369.5, 429.1)	(423.7, 481.0)	(471.7, 522.0)	(523.3, 557.4)	(590.7, 592.6)
			$n = 100$			
0.01	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.02	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)	(∞, ∞)
0.05	(542.0, 622.5)	(588.2, 635.7)	(635.1, 645.8)	(647.2, 647.8)	(648.5, 648.0)	(648.5, 648.0)
	(542.4, 622.7)	(588.5, 635.8)	(635.2, 645.8)	(647.2, 647.8)	(648.5, 648.0)	(648.5, 648.0)
0.10	(423.4, 567.9)	(481.6, 586.5)	(566.5, 625.3)	(613.1, 639.8)	(639.7, 646.6)	(650.2, 649.7)
	(423.9, 568.2)	(482.0, 586.8)	(566.9, 625.5)	(613.3, 639.9)	(639.8, 646.6)	(650.2, 649.7)
0.15	(340.5, 414.0)	(384.1, 440.6)	(418.1, 449.2)	(415.7, 430.6)	(405.0, 409.3)	(398.2, 397.7)
	(340.8, 414.2)	(384.4, 440.8)	(418.2, 449.2)	(415.7, 430.5)	(404.9, 409.2)	(398.2, 397.7)
0.20	(327.1, 385.5)	(364.6, 409.9)	(401.9, 429.7)	(419.1, 433.0)	(423.1, 426.3)	(422.8, 422.3)
	(327.4, 385.7)	(364.9, 410.1)	(402.1, 429.8)	(419.2, 433.0)	(423.1, 426.3)	(422.8, 422.3)

Table 5: Exact and approximate (ARL_0, $SDRL_0$) for the hypergeometric np chart with $N = 500$, $n \in \{25, 50, 75, 100\}$, $p_0 \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$, $m \in \{10, 20, 50, 100, 200, 1000\}$, $K = 3$

p_0	$m = 10$	$m = 20$	$m = 50$	$m = 100$	$m = 200$	$m = 1000$
			$n = 25$			
0.01	(858.3, 35490.8)	(276.1, 1427.8)	(159.4, 457.9)	(105.8, 338.1)	(59.0, 166.2)	(45.7, 45.2)
	(851.8, 35047.1)	(275.7, 1416.7)	(159.2, 457.5)	(105.7, 337.6)	(59.0, 165.8)	(45.7, 45.2)
0.02	(577.9, 8149.3)	(234.1, 873.9)	(163.0, 416.0)	(110.2, 213.6)	(95.9, 105.7)	(95.0, 94.5)
	(575.9, 8001.7)	(233.7, 869.8)	(162.8, 415.5)	(110.1, 213.1)	(95.9, 105.6)	(95.0, 94.5)
0.05	(502.9, 4239.4)	(295.0, 828.7)	(216.3, 395.0)	(183.8, 230.2)	(179.9, 182.2)	(180.2, 179.7)
	(501.5, 4202.3)	(294.7, 826.1)	(216.1, 394.5)	(183.8, 229.9)	(179.9, 182.1)	(180.2, 179.7)
0.10	(675.0, 3470.1)	(405.7, 995.0)	(306.2, 471.6)	(285.7, 426.6)	(259.7, 397.1)	(173.9, 261.4)
	(673.6, 3450.3)	(405.3, 992.7)	(306.1, 471.3)	(285.6, 426.6)	(259.6, 396.9)	(173.7, 261.2)
0.15	(737.4, 3123.5)	(481.2, 1065.6)	(384.0, 565.2)	(374.2, 502.9)	(363.8, 493.0)	(338.5, 470.4)
	(736.1, 3110.3)	(480.7, 1063.2)	(383.9, 564.7)	(374.2, 502.9)	(363.8, 493.0)	(338.5, 470.3)
0.20	(665.6, 1535.0)	(554.5, 1124.2)	(452.5, 670.0)	(416.6, 592.9)	(373.7, 538.6)	(257.2, 326.7)
	(665.3, 1534.0)	(554.1, 1122.7)	(452.3, 669.5)	(416.5, 592.8)	(373.6, 538.4)	(257.1, 326.4)
			$n = 50$			
0.01	(1689.2, 16206.5)	(425.9, 3125.3)	(209.3, 674.2)	(135.0, 282.0)	(122.7, 128.9)	(122.4, 121.9)
	(1681.6, 16152.2)	(424.9, 3104.7)	(208.9, 673.0)	(134.9, 281.1)	(122.7, 128.8)	(122.4, 121.9)
0.02	(762.5, 17279.5)	(371.3, 1357.2)	(268.5, 496.3)	(216.0, 419.7)	(160.8, 330.8)	(87.7, 108.5)
	(759.8, 16946.2)	(370.8, 1351.5)	(268.4, 496.0)	(215.9, 419.5)	(160.6, 330.5)	(87.6, 108.3)
0.05	(685.7, 4658.6)	(407.3, 1040.3)	(324.6, 495.7)	(297.5, 439.1)	(267.4, 406.2)	(177.3, 268.8)
	(684.1, 4621.2)	(406.9, 1037.5)	(324.5, 495.4)	(297.5, 439.0)	(267.3, 406.1)	(177.2, 268.6)
0.10	(722.3, 2025.9)	(508.6, 1078.0)	(396.9, 569.9)	(379.4, 475.4)	(388.9, 475.7)	(427.0, 499.9)
	(721.5, 2022.6)	(508.2, 1075.9)	(396.8, 569.4)	(379.4, 475.3)	(388.9, 475.7)	(427.1, 500.0)
0.15	(448.5, 747.3)	(441.0, 652.1)	(406.5, 536.2)	(375.5, 469.8)	(337.6, 400.3)	(289.6, 290.8)
	(448.6, 747.2)	(441.0, 651.8)	(406.5, 535.9)	(375.4, 469.7)	(337.5, 400.0)	(289.6, 290.8)
0.20	(366.1, 476.2)	(377.2, 463.8)	(386.7, 446.9)	(392.9, 448.1)	(409.8, 468.7)	(488.7, 549.2)
	(366.1, 476.2)	(377.3, 463.7)	(386.7, 446.8)	(393.0, 448.2)	(409.9, 468.8)	(488.8, 549.3)
			$n = 75$			
0.01	(1179.2, 5019.0)	(635.5, 3060.3)	(362.0, 833.4)	(369.7, 461.4)	(401.0, 465.1)	(470.4, 477.0)
	(1177.0, 5012.7)	(634.1, 3053.1)	(361.9, 830.2)	(369.8, 461.3)	(401.1, 465.1)	(470.4, 477.0)
0.02	(1184.2, 50250.6)	(486.4, 1810.9)	(345.0, 623.3)	(309.7, 550.8)	(250.6, 475.1)	(128.9, 208.9)
	(1179.6, 49415.1)	(485.7, 1803.1)	(344.9, 622.8)	(309.6, 550.7)	(250.5, 474.8)	(128.8, 208.5)
0.05	(919.6, 5555.2)	(516.4, 1270.3)	(389.6, 597.9)	(374.5, 504.9)	(371.5, 499.4)	(341.9, 473.5)
	(917.6, 5521.4)	(515.9, 1266.9)	(389.5, 597.3)	(374.5, 504.8)	(371.4, 499.4)	(341.8, 473.4)
0.10	(448.7, 747.3)	(456.7, 704.2)	(417.4, 553.2)	(380.8, 478.5)	(340.2, 405.2)	(289.8, 291.4)
	(448.7, 747.2)	(456.7, 703.9)	(417.4, 553.0)	(380.7, 478.4)	(340.1, 405.0)	(289.8, 291.4)
0.15	(375.5, 485.7)	(390.2, 480.9)	(405.4, 483.9)	(409.4, 484.2)	(397.8, 469.1)	(324.6, 351.9)
	(375.6, 485.7)	(390.3, 480.9)	(405.4, 483.9)	(409.4, 484.2)	(397.7, 469.0)	(324.5, 351.7)
0.20	(329.2, 409.1)	(368.3, 438.6)	(389.5, 444.7)	(388.6, 426.6)	(375.3, 394.5)	(355.6, 355.3)
	(329.2, 409.2)	(368.4, 438.7)	(389.5, 444.6)	(388.6, 426.5)	(375.3, 394.4)	(355.6, 355.3)
			$n = 100$			
0.01	(1017.6, 2302.8)	(981.7, 2240.0)	(535.2, 1570.9)	(328.6, 1081.4)	(196.7, 550.4)	(155.3, 154.8)
	(1017.4, 2302.5)	(981.2, 2239.4)	(534.4, 1569.5)	(328.0, 1079.7)	(196.4, 548.9)	(155.3, 154.8)
0.02	(1854.5, 58925.7)	(677.3, 3072.1)	(437.4, 852.9)	(355.2, 696.1)	(258.8, 504.6)	(171.9, 177.8)
	(1845.7, 58349.4)	(676.2, 3053.8)	(437.1, 852.0)	(354.9, 695.6)	(258.6, 504.0)	(171.9, 177.7)
0.05	(982.9, 3235.6)	(655.7, 1562.8)	(474.5, 722.2)	(428.3, 607.7)	(388.4, 557.8)	(265.6, 347.6)
	(981.7, 3229.7)	(655.1, 1559.4)	(474.4, 721.5)	(428.2, 607.5)	(388.2, 557.7)	(265.5, 347.3)
0.10	(386.2, 520.0)	(384.1, 480.7)	(388.5, 451.4)	(396.2, 453.3)	(413.7, 474.2)	(489.9, 551.7)
	(386.3, 519.9)	(384.1, 480.6)	(388.5, 451.4)	(396.2, 453.4)	(413.7, 474.3)	(490.0, 551.8)
0.15	(330.5, 410.0)	(367.6, 438.9)	(390.9, 446.6)	(390.6, 430.5)	(377.0, 397.7)	(355.7, 355.4)
	(330.6, 410.1)	(367.7, 439.0)	(390.9, 446.6)	(390.5, 430.4)	(376.9, 397.6)	(355.7, 355.4)
0.20	(317.7, 373.2)	(348.9, 392.6)	(375.1, 407.5)	(376.0, 401.0)	(362.3, 381.9)	(319.5, 323.2)
	(317.8, 373.3)	(349.0, 392.6)	(375.1, 407.5)	(376.0, 400.9)	(362.2, 381.8)	(319.5, 323.1)

Table 6: Exact and approximate (ARL_0, $SDRL_0$) for the hypergeometric np chart with $N = 1000$, $n \in \{25, 50, 75, 100\}$, $p_0 \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$, $m \in \{10, 20, 50, 100, 200, 1000\}$, $K = 3$

p_0	$m = 10$	$m = 20$	$m = 50$	$m = 100$	$m = 200$	$m = 1000$
			$n = 25$			
0.01	(426.1, 11240.4)	(216.9, 1083.9)	(122.2, 323.3)	(85.0, 241.3)	(55.6, 141.8)	(41.9, 41.4)
	(425.0, 11105.3)	(216.7, 1080.9)	(122.1, 323.1)	(85.0, 241.2)	(55.6, 141.7)	(41.9, 41.4)
0.02	(430.8, 5780.5)	(253.7, 825.8)	(135.4, 316.1)	(100.8, 191.8)	(85.6, 98.0)	(84.1, 83.6)
	(430.0, 5745.7)	(253.5, 824.1)	(135.4, 315.9)	(100.7, 191.6)	(85.6, 97.9)	(84.1, 83.6)
0.05	(556.3, 3064.8)	(301.2, 790.4)	(197.7, 357.2)	(165.5, 216.9)	(158.1, 162.2)	(157.6, 157.1)
	(555.7, 3054.4)	(301.1, 789.3)	(197.7, 357.0)	(165.5, 216.8)	(158.1, 162.2)	(157.6, 157.1)
0.10	(569.5, 3120.9)	(371.7, 848.9)	(295.9, 439.5)	(277.5, 392.0)	(266.5, 381.2)	(219.0, 329.3)
	(568.9, 3111.0)	(371.5, 847.9)	(295.9, 439.3)	(277.5, 392.0)	(266.5, 381.2)	(219.0, 329.2)
0.15	(745.7, 3323.5)	(460.6, 995.3)	(363.7, 525.5)	(360.9, 459.7)	(373.1, 465.4)	(411.7, 488.5)
	(745.0, 3315.7)	(460.4, 994.3)	(363.7, 525.2)	(360.9, 459.7)	(373.1, 465.4)	(411.7, 488.5)
0.20	(756.6, 2072.3)	(532.1, 1086.2)	(444.1, 642.4)	(414.6, 563.4)	(394.5, 542.2)	(315.7, 447.1)
	(756.3, 2071.0)	(531.9, 1085.4)	(444.1, 642.1)	(414.6, 563.3)	(394.5, 542.1)	(315.6, 447.0)
			$n = 50$			
0.01	(554.2, 12310.4)	(227.0, 1027.7)	(152.4, 377.4)	(104.8, 196.4)	(92.0, 103.8)	(90.8, 90.3)
	(553.0, 12194.3)	(226.8, 1025.1)	(152.3, 377.1)	(104.7, 196.2)	(92.0, 103.7)	(90.8, 90.3)
0.02	(577.4, 5163.4)	(293.1, 837.7)	(201.3, 339.3)	(187.4, 311.5)	(167.8, 289.5)	(97.0, 172.9)
	(576.6, 5139.2)	(293.0, 836.4)	(201.2, 339.2)	(187.3, 311.5)	(167.7, 289.5)	(96.9, 172.8)
0.05	(586.6, 3088.8)	(373.9, 865.0)	(277.1, 405.5)	(267.1, 351.9)	(265.8, 349.1)	(275.4, 356.0)
	(586.0, 3078.5)	(373.7, 864.0)	(277.0, 405.3)	(267.1, 351.9)	(265.8, 349.1)	(275.4, 356.1)
0.10	(708.3, 2450.1)	(451.6, 957.4)	(357.7, 512.0)	(342.4, 402.9)	(356.5, 387.4)	(401.0, 404.4)
	(707.7, 2447.0)	(451.4, 956.3)	(357.7, 511.8)	(342.4, 402.9)	(356.5, 387.5)	(401.0, 404.4)
0.15	(449.6, 709.9)	(422.8, 622.7)	(376.0, 479.3)	(359.0, 436.7)	(341.2, 415.9)	(279.0, 334.0)
	(449.6, 709.9)	(422.7, 622.6)	(376.0, 479.2)	(359.0, 436.6)	(341.2, 415.9)	(278.9, 333.9)
0.20	(364.5, 461.3)	(375.6, 445.2)	(390.2, 437.3)	(407.9, 443.5)	(435.5, 458.4)	(477.3, 478.2)
	(364.5, 461.3)	(375.7, 445.2)	(390.2, 437.3)	(408.0, 443.6)	(435.5, 458.5)	(477.3, 478.2)
			$n = 75$			
0.01	(781.1, 13598.1)	(273.6, 1150.0)	(196.8, 367.3)	(200.5, 230.1)	(217.3, 227.9)	(230.4, 229.9)
	(779.5, 13477.1)	(273.4, 1147.1)	(196.8, 366.9)	(200.5, 230.1)	(217.3, 227.9)	(230.4, 229.9)
0.02	(629.9, 4665.0)	(346.2, 906.3)	(255.1, 386.7)	(248.4, 344.3)	(247.1, 342.8)	(245.5, 341.6)
	(629.2, 4643.4)	(346.0, 905.0)	(255.1, 386.5)	(248.4, 344.3)	(247.1, 342.8)	(245.5, 341.6)
0.05	(669.1, 3380.2)	(413.7, 899.6)	(318.5, 464.3)	(303.7, 359.7)	(321.3, 351.2)	(363.0, 366.2)
	(668.4, 3370.7)	(413.5, 898.5)	(318.5, 464.1)	(303.7, 359.7)	(321.3, 351.2)	(363.0, 366.2)
0.10	(475.2, 787.2)	(420.1, 627.8)	(365.5, 459.8)	(351.0, 410.4)	(348.5, 403.0)	(347.3, 401.5)
	(475.2, 787.1)	(420.1, 627.6)	(365.4, 459.7)	(351.0, 410.3)	(348.5, 403.0)	(347.3, 401.5)
0.15	(370.4, 510.0)	(383.2, 498.1)	(387.3, 464.9)	(379.1, 425.2)	(386.0, 416.1)	(431.2, 440.7)
	(370.5, 510.1)	(383.3, 498.1)	(387.2, 464.9)	(379.1, 425.1)	(386.0, 416.2)	(431.2, 440.7)
0.20	(338.8, 422.7)	(359.8, 421.4)	(364.9, 398.1)	(361.5, 381.5)	(359.9, 376.4)	(358.8, 374.5)
	(338.8, 422.7)	(359.9, 421.4)	(364.9, 398.1)	(361.5, 381.4)	(359.9, 376.4)	(358.8, 374.5)
			$n = 100$			
0.01	(694.0, 22713.8)	(345.1, 1221.0)	(252.0, 468.0)	(203.3, 390.0)	(161.3, 324.6)	(86.7, 117.2)
	(692.7, 22439.4)	(344.8, 1218.5)	(251.9, 467.8)	(203.2, 389.9)	(161.2, 324.5)	(86.6, 117.1)
0.02	(684.0, 5737.5)	(372.8, 1032.0)	(281.9, 438.9)	(277.8, 386.1)	(271.7, 380.3)	(261.1, 371.7)
	(683.0, 5711.7)	(372.6, 1030.5)	(281.9, 438.7)	(277.8, 386.1)	(271.7, 380.3)	(261.1, 371.7)
0.05	(720.3, 2861.2)	(451.1, 973.3)	(357.8, 513.5)	(338.9, 404.4)	(354.0, 386.5)	(400.3, 404.1)
	(719.7, 2856.2)	(451.0, 972.2)	(357.7, 513.2)	(338.9, 404.3)	(354.0, 386.5)	(400.3, 404.1)
0.10	(372.5, 508.1)	(392.6, 516.0)	(390.5, 486.0)	(373.3, 436.9)	(352.8, 381.9)	(335.1, 334.7)
	(372.6, 508.1)	(392.7, 516.0)	(390.5, 486.0)	(373.2, 436.8)	(352.8, 381.8)	(335.1, 334.7)
0.15	(339.0, 419.5)	(356.1, 418.6)	(370.0, 412.4)	(382.5, 417.2)	(403.1, 434.4)	(469.8, 483.8)
	(339.1, 419.5)	(356.2, 418.6)	(370.0, 412.4)	(382.5, 417.2)	(403.1, 434.4)	(469.8, 483.8)
0.20	(312.6, 363.1)	(339.1, 374.4)	(363.7, 384.4)	(378.7, 393.3)	(394.1, 403.1)	(415.7, 415.5)
	(312.6, 363.2)	(339.1, 374.4)	(363.7, 384.4)	(378.7, 393.3)	(394.1, 403.1)	(415.7, 415.5)

Table 7: Exact and approximate (ARL_0, SDRL_0) for the hypergeometric np chart with $N = 2000$, $n \in \{25, 50, 75, 100\}$, $p_0 \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$, $m \in \{10, 20, 50, 100, 200, 1000\}$, $K = 3$

p_0	$m = 10$	$m = 20$	$m = 50$	$m = 100$	$m = 200$	$m = 1000$
			$n = 25$			
0.01	(340.7, 5663.0)	(188.3, 826.8)	(109.9, 280.2)	(77.9, 210.2)	(52.4, 125.1)	(40.3, 39.8)
	(340.3, 5635.8)	(188.2, 825.8)	(109.9, 280.1)	(77.9, 210.1)	(52.4, 125.0)	(40.3, 39.8)
0.02	(369.4, 4129.8)	(227.1, 695.8)	(124.9, 280.7)	(94.5, 172.6)	(81.4, 95.4)	(79.6, 79.1)
	(369.1, 4118.4)	(227.0, 695.2)	(124.9, 280.6)	(94.5, 172.5)	(81.4, 95.4)	(79.6, 79.1)
0.05	(499.2, 2922.3)	(276.2, 702.1)	(186.6, 325.6)	(158.5, 212.1)	(149.1, 154.6)	(148.1, 147.6)
	(498.9, 2916.7)	(276.2, 701.6)	(186.6, 325.5)	(158.5, 212.0)	(149.1, 154.6)	(148.1, 147.6)
0.10	(591.4, 2783.5)	(367.6, 845.6)	(290.9, 425.5)	(277.9, 379.3)	(268.7, 370.9)	(246.2, 350.4)
	(591.1, 2779.4)	(367.5, 845.1)	(290.9, 425.4)	(277.9, 379.2)	(268.7, 370.9)	(246.2, 350.4)
0.15	(688.4, 2955.0)	(459.3, 1018.2)	(354.3, 508.4)	(353.4, 440.0)	(370.3, 446.3)	(425.8, 474.5)
	(688.1, 2951.6)	(459.2, 1017.7)	(354.3, 508.3)	(353.4, 440.0)	(370.4, 446.3)	(425.8, 474.5)
0.20	(697.4, 1872.1)	(546.5, 1106.4)	(431.5, 618.7)	(417.6, 550.8)	(410.7, 543.3)	(366.4, 501.0)
	(697.3, 1871.5)	(546.4, 1106.0)	(431.4, 618.5)	(417.6, 550.8)	(410.7, 543.3)	(366.4, 501.0)
			$n = 50$			
0.01	(382.0, 4557.7)	(232.5, 722.3)	(127.1, 288.0)	(95.8, 176.5)	(82.3, 96.9)	(80.5, 80.0)
	(381.7, 4543.6)	(232.4, 721.7)	(127.0, 287.9)	(95.8, 176.4)	(82.3, 96.8)	(80.5, 80.0)
0.02	(442.2, 3151.6)	(243.4, 669.4)	(187.8, 299.7)	(182.5, 279.9)	(166.6, 265.0)	(118.8, 206.7)
	(441.9, 3144.9)	(243.4, 668.8)	(187.8, 299.6)	(182.5, 279.9)	(166.6, 265.0)	(118.8, 206.7)
0.05	(543.8, 2804.1)	(321.6, 747.7)	(254.2, 373.6)	(252.0, 316.6)	(263.9, 321.0)	(300.3, 340.0)
	(543.5, 2800.1)	(321.5, 747.2)	(254.1, 373.5)	(252.0, 316.6)	(263.9, 321.0)	(300.3, 340.0)
0.10	(636.9, 1991.2)	(426.5, 880.7)	(339.7, 475.9)	(320.7, 375.1)	(329.0, 346.6)	(353.1, 353.0)
	(636.7, 1990.0)	(426.5, 880.2)	(339.7, 475.8)	(320.7, 375.0)	(329.0, 346.6)	(353.1, 353.0)
0.15	(428.8, 707.1)	(407.7, 605.2)	(366.9, 459.1)	(352.6, 414.7)	(350.4, 409.5)	(337.3, 398,3)
	(428.8, 707.0)	(407.7, 605.1)	(366.9, 459.0)	(352.6, 414.7)	(350.4, 409.5)	(337.3, 398,3)
0.20	(364.9, 482.9)	(368.1, 447.1)	(390.6, 455.4)	(395.5, 434.2)	(405.4, 420.5)	(419.5, 419.0)
	(364.9, 482.9)	(368.1, 447.1)	(390.6, 455.4)	(395.5, 434.2)	(405.4, 420.5)	(419.5, 419.0)
			$n = 75$			
0.01	(452.7, 4330.7)	(250.1, 728.7)	(164.8, 286.7)	(163.6, 182.8)	(172.4, 177.1)	(178.6, 178.1)
	(452.3, 4319.3)	(250.0, 728.1)	(164.7, 286.6)	(163.6, 182.8)	(172.4, 177.1)	(178.6, 178.1)
0.02	(487.0, 2977.2)	(290.4, 728.1)	(214.4, 321.2)	(219.1, 280.3)	(227.9, 283.7)	(265.8, 301.1)
	(486.7, 2971.0)	(290.3, 727.6)	(214.4, 321.1)	(219.1, 280.3)	(227.9, 283.7)	(265.8, 301.1)
0.05	(602.5, 2618.7)	(367.6, 789.1)	(288.2, 414.1)	(270.7, 319.1)	(278.5, 291.6)	(295.3, 294.9)
	(602.2, 2615.0)	(367.5, 788.7)	(288.2, 414.0)	(270.7, 319.0)	(278.5, 291.6)	(295.3, 294.9)
0.10	(455.6, 747.4)	(399.7, 581.4)	(356.8, 440.9)	(352.0, 403.4)	(367.2, 407.0)	(422.1, 438.1)
	(455.6, 747.3)	(399.7, 581.3)	(356.8, 440.9)	(352.0, 403.3)	(367.2, 407.0)	(422.1, 438.1)
0.15	(374.9, 516.2)	(381.7, 491.2)	(366.9, 436.7)	(354.9, 392.6)	(353.5, 367.8)	(366.9, 366.8)
	(374.9, 516.2)	(381.7, 491.2)	(366.9, 436.7)	(354.9, 392.6)	(353.5, 367.8)	(366.9, 366.8)
0.20	(331.8, 408.7)	(361.4, 421.7)	(371.2, 411.9)	(384.8, 418.2)	(409.0, 437.6)	(472.8, 481.6)
	(331.8, 408.7)	(361.4, 421.7)	(371.2, 411.9)	(384.8, 418.2)	(409.0, 437.6)	(472.8, 481.6)
			$n = 100$			
0.01	(524.0, 4395.5)	(274.1, 756.1)	(209.0, 339.4)	(189.3, 303.3)	(166.7, 279.7)	(104.1, 187.2)
	(523.6, 4384.9)	(274.0, 755.6)	(209.0, 339.4)	(189.3, 303.2)	(166.7, 279.7)	(104.1, 187.2)
0.02	(523.6, 3216.3)	(299.2, 726.4)	(240.6, 354.0)	(236.7, 296.4)	(251.6, 301.2)	(269.7, 320.6)
	(523.3, 3210.2)	(299.2, 725.9)	(240.6, 353.9)	(236.7, 296.4)	(251.6, 301.2)	(269.7, 320.6)
0.05	(664.2, 2495.3)	(420.5, 876.7)	(323.4, 460.8)	(297.2, 355.4)	(293.6, 307.5)	(302.2, 301.7)
	(663.9, 2493.2)	(420.4, 876.2)	(323.3, 460.7)	(297.1, 355.4)	(293.6, 307.5)	(302.2, 301.7)
0.10	(400.0, 597.2)	(390.3, 544.9)	(363.2, 452.5)	(346.2, 405.6)	(325.9, 374.3)	(270.0, 283.4)
	(400.0, 597.2)	(390.3, 544.9)	(363.2, 452.5)	(346.1, 405.6)	(325.9, 374.3)	(270.0, 283.4)
0.15	(336.5, 421.8)	(356.1, 427.1)	(369.8, 420.2)	(374.4, 410.7)	(375.4, 393.7)	(378.3, 377.9)
	(336.5, 421.8)	(356.1, 427.1)	(369.8, 420.2)	(374.4, 410.7)	(375.4, 393.7)	(378.3, 377.9)
0.20	(315.1, 376.4)	(348.0, 396.4)	(367.3, 403.5)	(371.8, 400.2)	(360.8, 381.5)	(323.8, 326.7)
	(315.1, 376.4)	(348.0, 396.4)	(367.3, 403.5)	(371.8, 400.2)	(360.8, 381.5)	(323.8, 326.7)

Table 8: Exact and approximate (ARL_0, SDRL_0) for the hypergeometric np chart with $N = 5000$, $n \in \{25, 50, 75, 100\}$, $p_0 \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$, $m \in \{10, 20, 50, 100, 200, 1000\}$, $K = 3$

p_0	$m = 10$	$m = 20$	$m = 50$	$m = 100$	$m = 200$	$m = 1000$
			$n = 25$			
0.01	(304.0, 4198.0)	(174.6, 719.4)	(135.5, 310.6)	(88.6, 228.3)	(54.8, 133.8)	(39.4, 38.9)
	(303.9, 4190.7)	(174.6, 719.1)	(135.4, 310.6)	(88.6, 228.3)	(54.8, 133.7)	(39.4, 38.9)
0.02	(339.8, 3464.3)	(213.4, 633.7)	(138.7, 305.8)	(96.4, 183.8)	(79.4, 96.1)	(77.1, 76.6)
	(339.6, 3460.6)	(213.4, 633.4)	(138.7, 305.8)	(96.4, 183.8)	(79.4, 96.1)	(77.1, 76.6)
0.05	(468.6, 2685.7)	(262.8, 656.3)	(191.0, 338.6)	(156.1, 214.6)	(144.4, 152.1)	(142.9, 142.4)
	(468.5, 2683.6)	(262.8, 656.1)	(191.0, 338.5)	(156.1, 214.5)	(144.4, 152.1)	(142.9, 142.4)
0.10	(606.6, 2832.2)	(353.2, 834.1)	(279.0, 406.1)	(275.8, 369.9)	(270.9, 365.2)	(262.2, 358.1)
	(606.4, 2830.7)	(353.1, 833.9)	(279.0, 406.1)	(275.8, 369.9)	(270.9, 365.2)	(262.2, 358.1)
0.15	(680.2, 2760.5)	(457.2, 968.4)	(354.7, 507.6)	(347.0, 427.2)	(365.6, 433.4)	(427.6, 462.5)
	(680.1, 2759.3)	(457.2, 968.2)	(354.7, 507.6)	(347.0, 427.2)	(365.6, 433.4)	(427.6, 462.5)
0.20	(665.3, 1764.4)	(523.6, 1051.7)	(430.8, 614.4)	(420.5, 544.1)	(415.0, 537.4)	(396.7, 522.0)
	(665.2, 1764.2)	(523.6, 1051.6)	(430.8, 614.3)	(420.5, 544.1)	(415.0, 537.4)	(396.7, 522.0)
			$n = 50$			
0.01	(319.9, 3083.6)	(204.3, 592.5)	(134.0, 291.5)	(93.9, 176.0)	(77.7, 93.3)	(75.4, 74.9)
	(319.8, 3080.3)	(204.3, 592.3)	(134.0, 291.5)	(93.9, 176.0)	(77.6, 93.3)	(75.4, 74.9)
0.02	(384.6, 2636.8)	(219.3, 575.1)	(187.3, 287.8)	(177.2, 261.3)	(167.1, 252.6)	(133.8, 218.7)
	(384.5, 2634.4)	(219.2, 574.9)	(187.3, 287.8)	(177.2, 261.3)	(167.1, 252.6)	(133.8, 218.6)
0.05	(538.2, 2574.5)	(329.2, 711.7)	(245.7, 357.6)	(244.2, 298.7)	(254.8, 300.6)	(299.0, 320.9)
	(538.1, 2572.9)	(329.1, 711.6)	(245.7, 357.6)	(244.2, 298.7)	(254.8, 300.6)	(299.0, 320.9)
0.10	(659.9, 2253.3)	(427.8, 874.3)	(331.1, 468.3)	(308.7, 362.2)	(311.8, 325.4)	(326.7, 326.3)
	(659.8, 2252.7)	(427.8, 874.1)	(331.1, 468.2)	(308.7, 362.1)	(311.8, 325.4)	(326.7, 326.3)
0.15	(451.0, 725.3)	(407.8, 601.2)	(365.2, 455.3)	(356.0, 416.8)	(364.3, 421.3)	(399.1, 456.2)
	(451.0, 725.3)	(407.8, 601.2)	(365.2, 455.3)	(356.0, 416.8)	(364.3, 421.3)	(399.1, 456.2)
0.20	(370.3, 494.8)	(383.6, 492.7)	(393.0, 476.6)	(392.9, 446.2)	(387.7, 405.8)	(388.6, 388.1)
	(370.3, 494.8)	(383.6, 492.7)	(393.0, 476.6)	(392.9, 446.2)	(387.7, 405.8)	(388.6, 388.1)
			$n = 75$			
0.01	(354.0, 2707.6)	(213.5, 641.3)	(155.9, 268.2)	(146.9, 166.0)	(152.8, 155.5)	(156.6, 156.1)
	(353.9, 2705.0)	(213.5, 641.0)	(155.9, 268.2)	(146.9, 166.0)	(152.8, 155.5)	(156.6, 156.1)
0.02	(412.1, 2470.8)	(267.1, 636.7)	(199.8, 292.9)	(205.0, 251.3)	(216.8, 254.8)	(255.1, 269.3)
	(412.0, 2469.1)	(267.0, 636.5)	(199.8, 292.9)	(205.0, 251.3)	(216.8, 254.8)	(255.1, 269.3)
0.05	(536.0, 2326.6)	(356.2, 743.7)	(275.5, 392.8)	(255.1, 303.3)	(255.0, 264.6)	(261.9, 261.5)
	(535.9, 2325.4)	(356.1, 743.5)	(275.5, 392.8)	(255.0, 303.2)	(255.0, 264.6)	(261.9, 261.5)
0.10	(418.8, 685.9)	(380.8, 543.9)	(346.6, 421.7)	(341.9, 384.6)	(353.6, 378.5)	(392.6, 395.6)
	(418.8, 685.9)	(380.8, 543.8)	(346.6, 421.7)	(341.9, 384.6)	(353.6, 378.5)	(392.6, 395.6)
0.15	(361.7, 499.9)	(370.6, 483.0)	(363.0, 427.4)	(351.6, 387.5)	(345.0, 359.4)	(340.4, 341.4)
	(361.7, 499.9)	(370.6, 483.0)	(363.0, 427.4)	(351.6, 387.5)	(345.1, 359.4)	(340.4, 341.4)
0.20	(334.5, 407.8)	(357.0, 409.2)	(368.4, 401.3)	(382.8, 406.4)	(400.7, 416.4)	(434.4, 435.1)
	(334.5, 407.8)	(357.0, 409.2)	(368.4, 401.3)	(382.8, 406.4)	(400.7, 416.4)	(434.4, 435.1)
			$n = 100$			
0.01	(384.6, 2636.8)	(243.3, 668.9)	(187.3, 287.8)	(177.2, 261.4)	(167.1, 252.6)	(133.8, 218.7)
	(384.5, 2634.4)	(243.2, 668.7)	(187.3, 287.8)	(177.2, 261.4)	(167.1, 252.6)	(133.8, 218.6)
0.02	(452.4, 2273.7)	(284.5, 649.9)	(216.0, 320.2)	(215.6, 259.2)	(231.0, 259.9)	(267.6, 272.8)
	(452.3, 2272.3)	(284.5, 649.7)	(216.0, 320.2)	(215.6, 259.2)	(231.0, 259.9)	(267.6, 272.8)
0.05	(585.4, 2292.5)	(383.8, 774.8)	(304.2, 427.0)	(278.9, 342.7)	(262.3, 280.8)	(258.1, 257.6)
	(585.3, 2291.5)	(383.8, 774.6)	(304.2, 427.0)	(278.9, 342.6)	(262.3, 280.8)	(258.1, 257.6)
0.10	(391.2, 571.8)	(378.6, 526.1)	(349.0, 428.0)	(335.5, 385.1)	(326.2, 370.3)	(288.1, 328.4)
	(391.2, 571.8)	(378.6, 526.1)	(348.9, 428.0)	(335.5, 385.1)	(326.2, 370.3)	(288.1, 328.3)
0.15	(334.1, 420.4)	(352.7, 422.7)	(367.2, 422.7)	(365.4, 407.8)	(352.1, 377.4)	(325.7, 326.0)
	(334.1, 420.4)	(352.7, 422.7)	(367.2, 422.7)	(365.4, 407.8)	(352.1, 377.4)	(325.7, 326.0)
0.20	(323.4, 386.7)	(347.3, 396.6)	(363.8, 396.8)	(363.1, 385.1)	(355.8, 373.9)	(320.9, 337.6)
	(323.4, 386.7)	(347.3, 396.6)	(363.8, 396.8)	(363.1, 385.1)	(355.8, 373.9)	(320.9, 337.6)

Table 9: Exact and approximate (ARL_0, SDRL_0) for the hypergeometric np chart with $N = 10000$, $n \in \{25, 50, 75, 100\}$, $p_0 \in \{0.01, 0.02, 0.05, 0.10, 0.15, 0.20\}$, $m \in \{10, 20, 50, 100, 200, 1000\}$, $K = 3$

p_0	$m = 10$	$m = 20$	$m = 50$	$m = 100$	$m = 200$	$m = 1000$
			$n = 25$			
0.01	(293.4, 3844.4)	(170.5, 689.0)	(132.7, 302.8)	(87.1, 222.8)	(54.2, 130.8)	(39.1, 38.7)
	(293.4, 3841.1)	(170.5, 688.8)	(132.7, 302.8)	(87.1, 222.8)	(54.2, 130.8)	(39.1, 38.7)
0.02	(330.8, 3278.9)	(209.2, 615.0)	(136.6, 299.1)	(95.3, 180.2)	(78.6, 94.8)	(76.3, 75.8)
	(330.8, 3277.2)	(209.2, 614.9)	(136.5, 299.1)	(95.3, 180.2)	(78.6, 94.8)	(76.3, 75.8)
0.05	(459.1, 2595.6)	(258.6, 642.0)	(188.4, 332.6)	(154.3, 211.3)	(142.7, 150.2)	(141.2, 140.7)
	(459.0, 2594.6)	(258.6, 642.0)	(188.4, 332.5)	(154.3, 211.3)	(142.7, 150.2)	(141.2, 140.7)
0.10	(595.8, 2761.4)	(348.0, 818.4)	(277.0, 410.7)	(272.1, 364.7)	(273.6, 364.8)	(267.2, 359.8)
	(595.7, 2760.7)	(347.9, 818.3)	(277.0, 410.7)	(272.1, 364.7)	(273.6, 364.8)	(267.2, 359.8)
0.15	(669.8, 2770.7)	(450.6, 951.7)	(350.0, 500.2)	(349.7, 425.6)	(365.6, 430.0)	(427.1, 458.2)
	(669.7, 2770.1)	(450.6, 951.6)	(350.0, 500.2)	(349.7, 425.6)	(365.6, 430.0)	(427.2, 458.2)
0.20	(692.9, 1995.2)	(520.8, 1096.8)	(425.1, 605.5)	(414.9, 536.4)	(416.2, 535.2)	(406.4, 527.3)
	(692.9, 1995.0)	(520.8, 1096.7)	(425.1, 605.5)	(414.9, 536.4)	(416.2, 535.2)	(406.4, 527.3)
			$n = 50$			
0.01	(303.2, 2754.1)	(196.3, 565.6)	(129.8, 278.7)	(91.6, 169.1)	(76.7, 95.6)	(73.9, 73.4)
	(303.2, 2752.6)	(196.3, 565.5)	(129.8, 278.7)	(91.6, 169.1)	(76.7, 95.6)	(73.9, 73.4)
0.02	(368.1, 2433.0)	(234.9, 636.8)	(181.7, 277.9)	(172.0, 252.7)	(162.3, 244.3)	(138.9, 221.4)
	(368.0, 2432.0)	(234.9, 636.7)	(181.7, 277.9)	(172.0, 252.7)	(162.3, 244.3)	(138.9, 221.4)
0.05	(518.9, 2437.1)	(319.3, 685.0)	(239.1, 346.8)	(237.8, 291.3)	(251.7, 294.2)	(296.5, 314.0)
	(518.8, 2436.4)	(319.3, 684.9)	(239.1, 346.8)	(237.8, 291.3)	(251.7, 294.2)	(296.5, 314.0)
0.10	(638.6, 2157.5)	(415.8, 844.7)	(326.8, 455.6)	(305.0, 358.4)	(306.2, 318.9)	(318.4, 318.0)
	(638.5, 2157.2)	(415.8, 844.6)	(326.8, 455.6)	(305.0, 358.4)	(306.2, 318.9)	(318.4, 318.0)
0.15	(438.9, 704.5)	(397.4, 584.8)	(359.8, 451.4)	(355.7, 416.1)	(367.3, 423.7)	(415.4, 466.4)
	(438.9, 704.5)	(397.4, 584.7)	(359.8, 451.3)	(355.7, 416.1)	(367.3, 423.7)	(415.4, 466.4)
0.20	(361.1, 481.8)	(388.4, 498.3)	(397.2, 486.4)	(392.6, 449.3)	(381.7, 401.0)	(379.1, 378.6)
	(361.1, 481.8)	(388.4, 498.3)	(397.2, 486.4)	(392.6, 449.3)	(381.7, 401.0)	(379.1, 378.6)
			$n = 75$			
0.01	(329.1, 2356.4)	(202.2, 589.5)	(149.2, 252.6)	(141.0, 158.9)	(147.3, 149.4)	(150.3, 149.8)
	(329.1, 2355.4)	(202.2, 589.4)	(149.2, 252.5)	(141.0, 158.9)	(147.3, 149.4)	(150.3, 149.8)
0.02	(388.2, 2232.9)	(254.4, 596.1)	(194.3, 293.4)	(196.5, 241.0)	(211.5, 245.4)	(248.8, 258.9)
	(388.2, 2232.2)	(254.4, 596.0)	(194.3, 293.4)	(196.5, 241.0)	(211.5, 245.4)	(248.8, 258.9)
0.05	(514.8, 2334.0)	(341.5, 719.5)	(271.6, 391.4)	(251.2, 302.9)	(247.3, 257.2)	(252.1, 251.6)
	(514.8, 2333.4)	(341.5, 719.4)	(271.6, 391.4)	(251.2, 302.9)	(247.3, 257.2)	(252.1, 251.6)
0.10	(402.8, 657.8)	(368.5, 522.9)	(342.9, 414.9)	(337.4, 377.4)	(347.1, 368.1)	(379.9, 381.3)
	(402.8, 657.8)	(368.5, 522.9)	(342.9, 414.9)	(337.4, 377.4)	(347.1, 368.1)	(379.9, 381.3)
0.15	(376.8, 528.5)	(377.1, 489.8)	(358.4, 420.2)	(348.2, 382.8)	(342.1, 356.6)	(337.7, 338.9)
	(376.8, 528.5)	(377.1, 489.8)	(358.4, 420.2)	(348.2, 382.8)	(342.1, 356.6)	(337.7, 338.9)
0.20	(322.7, 392.7)	(349.4, 399.2)	(367.0, 396.2)	(378.7, 399.0)	(393.9, 406.2)	(419.2, 419.3)
	(322.7, 392.7)	(349.4, 399.2)	(367.0, 396.2)	(378.7, 399.0)	(393.9, 406.2)	(419.2, 419.3)
			$n = 100$			
0.01	(351.4, 2231.6)	(226.5, 604.5)	(176.0, 268.0)	(166.7, 244.0)	(164.4, 242.0)	(143.7, 223.0)
	(351.3, 2230.6)	(226.4, 604.4)	(176.0, 268.0)	(166.7, 244.0)	(164.4, 242.0)	(143.7, 223.0)
0.02	(420.8, 2293.0)	(271.5, 645.7)	(215.6, 318.5)	(209.0, 248.5)	(223.9, 247.6)	(255.9, 258.5)
	(420.8, 2292.2)	(271.5, 645.6)	(215.6, 318.5)	(209.0, 248.5)	(223.9, 247.6)	(255.9, 258.5)
0.05	(582.2, 2202.8)	(379.4, 777.6)	(298.6, 416.9)	(273.3, 337.6)	(253.7, 275.2)	(245.6, 245.1)
	(582.1, 2202.4)	(379.4, 777.5)	(298.6, 416.9)	(273.3, 337.6)	(253.7, 275.2)	(245.6, 245.1)
0.10	(386.8, 585.3)	(372.7, 512.1)	(346.1, 422.6)	(331.0, 376.8)	(325.6, 365.3)	(306.6, 347.5)
	(386.8, 585.3)	(372.7, 512.1)	(346.1, 422.6)	(331.0, 376.8)	(325.6, 365.3)	(306.6, 347.5)
0.15	(339.2, 422.9)	(354.2, 421.7)	(363.5, 416.3)	(360.5, 401.7)	(347.3, 375.1)	(311.9, 313.6)
	(339.2, 422.9)	(354.2, 421.7)	(363.5, 416.3)	(360.5, 401.7)	(347.3, 375.1)	(311.9, 313.6)
0.20	(322.9, 392.5)	(348.5, 398.7)	(360.2, 390.9)	(360.2, 379.1)	(355.6, 370.3)	(336.0, 352.5)
	(322.9, 392.5)	(348.5, 398.7)	(360.2, 390.9)	(360.2, 379.1)	(355.6, 370.3)	(336.0, 352.4)